CMPT 310 Artificial Intelligence Survey

Assignment 1 Report

Maoshun Shang

301280479

Instructor: James Delgrande

Search procedure:

The shortest path problem is solved by implementing an algorithm, which is a variant of Dijkstra's algorithm, called uniform-cost search. Because we do not have complete information about the whole diagram, which in this case, the shortest path from any non-terminal square to the goal state square. Therefore, the A* search cannot be used. Additionally, due to the cost of edges are not 1's, Breadth-first search is not optimal solution. Moreover, heuristic functions only give an estimate cost of shortest path, uniform-cost search is the best solution for giving out the complete shortest path from any given starting location to the goal state location.

How the program works:

The algorithm maintains two list of records:

1. A list named "visited squares" contains a list of squares that have been visited. It is used when exploring a new square, ensuring path does not turns back.
2. A priority queue named "fringe" storing tuples (cumulative cost, a specific path), where cumulative cost is an integer as the priority and a specific path is a list of square locations.

The program dequeue the first element, which is, a specific path and its cumulative cost with highest priority. Then, expands the path by visiting the neighbours of the last-known location along the path. Appending the neighbours to the path and update the cumulative cost. Then, put the modified dequeued element back into the "fringe".

The reason being for choosing the fringe as a priority queue is that, the priority queue automatically sorts the elements in the queue after insertion and updates, with smaller number as higher priority. Therefore, when dequeue the element with highest priority, it is guaranteed that such path is the shortest path the program has explored so far.

With the characteristics of priority queue, if the path to be dequeued has the goal state square as its last-known position at some time instant, this path is said to be the path we are looking for. Because it has the highest priority, and highest priority indicates the shortest path up to this last-known location in this particular path.

Time and space complexity:

Overall, the time and space complexity for this algorithm is $O(b^{ceil(C^*/e)})$, where the 'b' is the branching factor, 'C*' is the cost of the optimal solution and the 'e' stands for the lower bound for all the step cost. This is much more efficient than the operations done on brute-force solution: which is listing all the possible paths, evaluating each of them and find the least cost one.